HRM Digital I/O

Rapid turnaround matters

Mon, Mar 31, 2003

An HRM is billed as a possible replacement for an SRM in functionality of I/O analog and digital interfacing. In the current HRM 2 design, a Digital I/O module is used that has no processor on board. Of the three available module slots, support of digital I/O requires using one of those slots. This allows for support of a 10 KHz analog Slow Data module, a Digital I/O module, and a Snapshot (future) module, for example. This note describes some concerns that relate to digital I/O support.

Consider the case of a stepping motor that is interfaced to digital I/O lines. The current code simply drives the output lines directly, creating a 20 μs pulse, which is long enough to allow for some significant filtering to keep out noise. With the HRM, however, it may not be so simple, unless the HRM can replicate in the hardware very promptly the digital control actions performed by the CPU. So, how much delay occurs when writing to a digital output byte, and with how much jitter? When the write is made to a PMC register, the Hotlink message would have to be sent right away, and upon its reception, the remote motherboard would have to write to the affected digital output byte immediately.

If the above scheme cannot work, then it is necessary to call for a pulsed write action to the PMC register. A write to the register should cause a Hotlink transmission to the remote board, which would then see to it that a pulse is delivered to the output lines, using an appropriate delay, like 20 μs. During the pulse action, however, it should not be possible to write anything else to that digital output byte. Specifically, there could be PMC registers that are used only for writing. The data that is written to such registers implies that pulse support is needed. On reception of the HotLink message, the remote crate would write that data exclusive-ORed with the present data lines, await the pulse duration of 20 μs, or whatever, then restore what was there. What should the pulse duration actually be, and how much control over it can we have?

A worse problem may exist to support multiplexing of digital input data. To effect demultiplexing, some digital output lines are used as a multiplex selection, and digital input lines are read following each write to the output lines. For example, one can use this scheme to interface to 16 bytes of digital data by using 4 lines of control and 8 lines of input. The software just writes to the multiplex select lines, then reads back the digital input byte. Do this in a loop 16 times, and we have 16 bytes of digital input data, using only two bytes of digital I/O. To allow for enough time for the digital input lines to settle, the CPU just has to “know” how long to wait.